ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Title goes here

Master Thesis

Robin Bosshard, 16-915-399

October 16, 2023

Supervisors: Prof. Dr. Fernando Perez-Cruz, Dr. Eliza Harris, Lili Gasser (SDSC)
Dr. Kasia Arturi (Eawag)

Department of Computer Science, ETH Zürich

Abstract

Abstract goes here.

Contents

Chapter 1

Material and Methods

The ToxCast's *in vitro* database, along with its processing pipeline, is explored. These concepts are crucial for familiarizing ourselves with the procedure of generating the corresponding toxicity data from the HTS data used for constructing the underlying predictive machine learning models.

1.1 InvitroDB v4.1

The most recent release of the ToxCast's database[1], referred to as *invitroDBv4.1*, serves as a source of an extensive collection of HTS targeted bioactivity data. This database encompasses information on a total of 10 196 compounds, selectively screened across 1485 assay endpoints. The assays utilize a range of technologies to assess the impact of chemical exposure on a wide array of biological targets, including individual proteins and cellular processes such as mitochondrial health, developmental processes and nuclear receptor signaling.

This resource originated from the collaboration of two prominent institutions: the U.S. EPA through its ToxCast program and the *National Institutes of Health*[2] *(NIH)* via the Tox21 initiative. Utilizing data gathered from various research laboratories, this relational database is publicly available and can be downloaded[3] by visiting the official ToxCast website.

1.2 tcpl v3.0

The *tcpl*[4] package, written in R, offers a comprehensive suite of tools for managing HTS data, provides reproducible concentration-response modeling and populates the

[1]released on September 21, 2023
[2]https://ntp.niehs.nih.gov/whatwestudy/tox21
[3]https://www.epa.gov/chemical-research/exploring-toxcast-data
[4]https://github.com/USEPA/CompTox-ToxCast-tcpl

`MYSQL` database, *invitrodb*. The multiple-concentration screening paradigm intends to pinpoint the activity of compounds, while also estimating their efficacy and potency. The concentration-response modeling procedure also addresses outlier robustness and signal loss due to cytotoxicity. In Chapter 1, the `Python` re-implementation *pytcpl* of the fundamental components of the ToxCast pipeline tcpl is introduced but at this point the essential elements are layed out that underpin the entire pipeline.

Each compound-assay dataset involves the collection of the respective *concentration-response series (CRS)* denoted as S_{ij}, representing compound c_j in assay endpoint a_i. A CRS is represented as a set of concentration-response pairs:

$$S = \{(conc_1, resp_1), (conc_2, resp_2), \ldots, (conc_{n_{\text{datapoints}_{i,j}}}, resp_{n_{\text{datapoints}_{i,j}}})\}$$

where $n_{\text{datapoints}_{i,j}}$ varies based on the number of concentrations tested for compound c_j in the assay endpoint a_i.

The concentration-response pairs can be retrieved by combining tables *mc0*, *mc1*, and *mc3* from invitroDBv4.1, representing the raw data. Esential sample information, including well type and indices from the assay well-plate is also collected. The concentrations are transformed to the logarithmic scale having the unit μM (micromolar), while the responses are control well-normalized to either fold-induction or percent-of-control activity. A control well is a set of sample wells that serve as a baseline for comparison to the impact of the treated samples. Control wells typically contain untreated samples or samples with a known, non-toxic response. The control wells are used to normalize the treated samples to account for any background noise or variability in the assay. There are two methods for analyzing raw assay results that will impact the analysis of the background distribution [1]:

a. **Fold Induction:** Fold induction is a measure used to quantify how much a certain parameter (e.g., gene expression or protein activity) has changed in response to a treatment compared to its baseline level. For example, if a gene is expressed five times higher in a treated sample compared to the control, the fold induction would be 5.

a. **Percent of Control:** Percent of control is another way to express the relative change in a parameter due to treatment where the obervations range from 0 to a maximum value for both chemical-treated and control samples.

In practice, concentrations are often subjected to multiple testing iterations, resulting in the formation of distinct concentration groups with replicates. The following quantities are introduced corresponding to a single CRS given a compound c_i and assay endpoint a_i:

- $n_{\text{datapoints}_{i,j}}$: the total number of concentration-response pairs ($|S|$)
- $n_{\text{groups}_{i,j}}$: the number of distinct concentrations tested
- $n_{\text{replicates}_{i,j}}$: the number of replicates for each concentration group

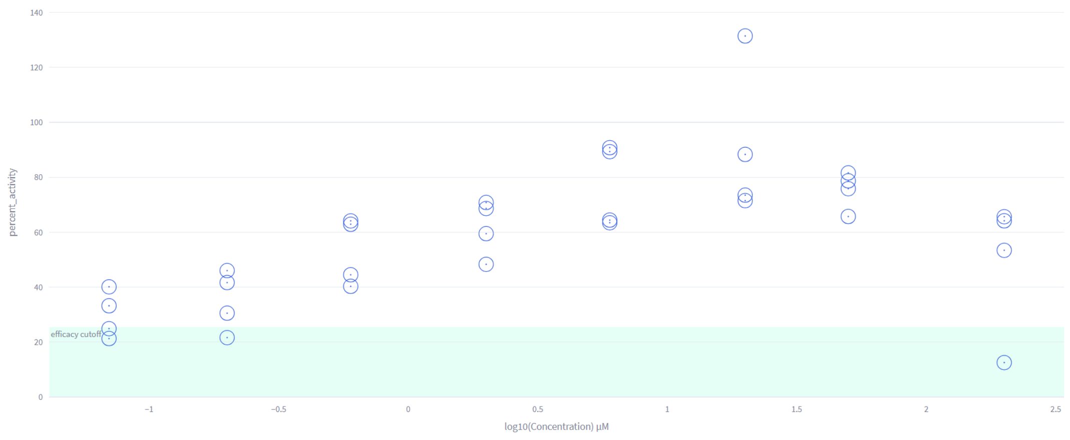


Figure 1.1: A CRS for the compound *Estropipate* (DTXSID3023005) in the assay endpoint *TOX21_ERa_LUC_VM7_Agonist* (aeid=788). The series has a total of $k = 45$ concentration-response pairs and is composed of $n_{conc} = 15$ concentration groups, each with $n_{rep} = 3$ replicates.

- $min_{\text{conc}_{i,j}}$: the lowest concentration tested
- $max_{\text{conc}_{i,j}}$: the highest concentration tested

Figure 1.1 showcases a single CRS for some compound tested within an assay endpoint.

To gain a rough visual representation of how these quantities vary across the complete set of analyzed concentration-response series in this thesis, please consult Figure 1.2. This figure illustrates the above metrics aggregated by their means, grouped by assay endpoints and compounds.

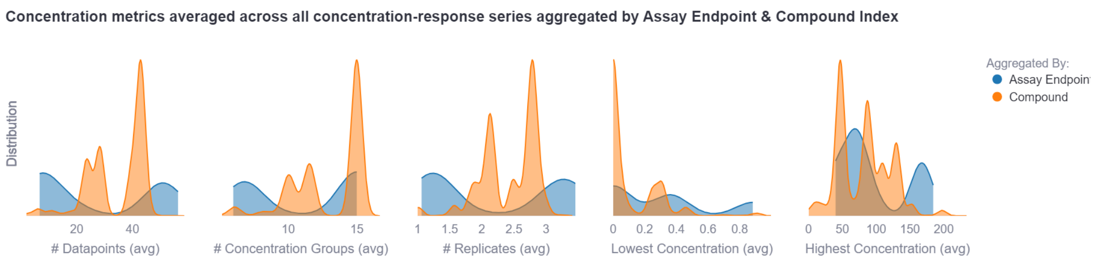


Figure 1.2: Concentration metrics averaged across all concentration-response series aggregated by assay endpoint (blue) and compound (orange). E.g., the first chart shows the distribution on the average number of datapoints across all assay enpoint $a_i \in A$ with $\frac{1}{|C|}\sum_j n_{\text{datapoints}_{i,j}}$ and across all compounds $c_j \in C$ with $\frac{1}{|A|}\sum_i n_{\text{datapoints}_{i,j}}$. Similarly, the process is repeated for the other metrics: $n_{\text{groups}_{i,j}}$, $n_{\text{replicates}_{i,j}}$, $min_{\text{conc}_{i,j}}$, and $max_{\text{conc}_{i,j}}$.

1.2.1 tcplFit2

To improve upon tcpl, `R` package *tcplFit2*[5] was developed, a standalone package focused on curve-fitting and hit-calling. The package also offers a more flexible and robust fitting procedure, allowing for the use of different optimization algorithms and the incorporation of user-defined constraints. *tcplFit2* differs from other `R`-language open-source concentration-response packages like *drc* and *mixtox*, as it is specifically tailored for HTS concentration-response data, offering an extensive set of curve models, summarized in Table 1.1.

Table 1.1: tcplfit2 Model Details

Model	Label	Equations[1]
Constant	cnst	$f(x) = 0$
Linear	poly1	$f(x) = ax$
Quadratic	poly2	$f(x) = a\left(\frac{x}{b} + \left(\frac{x}{b}\right)^2\right)$
Power	pow	$f(x) = ax^p$
Hill	hill	$f(x) = \frac{tp}{1+\left(\frac{ga}{x}\right)^p}$
Gain-Loss	gnls	$f(x) = \frac{tp}{(1+\left(\frac{ga}{x}\right)^p)(1+\left(\frac{x}{la}\right)^q)}$
Exponential 2	exp2	$f(x) = a\left(\exp\left(\frac{x}{b}\right) - 1\right)$
Exponential 3	exp3	$f(x) = a\left(\exp\left(\left(\frac{x}{b}\right)^p\right) - 1\right)$
Exponential 4	exp4	$f(x) = tp\left(1 - 2^{-\frac{x}{ga}}\right)$
Exponential 5	exp5	$f(x) = tp\left(1 - 2^{-\left(\frac{x}{ga}\right)^p}\right)$

[1] Model parameters: a: x-scale, b: y-scale p: (gain) power, q: (loss) power, tp: top, ga: gain AC50, la: loss AC50

All the curve fit models assume that the normalized observations from the CRS conform to a Student's t-distribution with 4 degrees of freedom [2] The Student's t-distribution has heavier tails compared to the normal distribution, making it more robust to outlier and eliminates the necessity of removing potential outliers prior to the fitting process. The model fitting algorithm in tcplFit2 employs nonlinear *maximum likelihood estimation (MLE)* to determine the model parameters for all available models.

Consider $t(z, \nu)$ as the Student's t-distribution with ν degrees of freedom, where y_i represents the observed response for the i-th observation, and μ_i is the estimated

[5] https://github.com/USEPA/CompTox-ToxCast-tcplFit2

response for the same observation. The calculation of z_i is as follows:

$$z_i = y_i - \mu_i \exp(\sigma)$$

where σ is the scale term.

Then the log-likelihood is

$$\sum_{i=1}^{n} [\ln(t(z_i, 4)) - \sigma]$$

where n is the number of observations.

The *Akaike Information Criterion (AIC)*[6] is used as as a measure of goodness of fit. The model with the lowest AIC value is chosen as the *winning* model. The winning model is then used to estimate the efficacy and potency of the compound. More precise, the potency estimates, also called *point-of-departure (POD)* estimates, are derived from the fitted curve characteristics, identifying *activity concentrations (AC)* at which the curve crosses certain response levels. For example:

- the AC at which the compound first reaches 50% of its estimated maximum response is denoted by *ac50*
- the AC at which the compound first reaches the estimated efficacy cutoff is denoted by *acc*
- the AC at which the compound first reaches the estimated assay background noise *3bmad*[7] is denoted by *acb*

Figure X illustrates the stated POD estimates. Notably, no POD estimates are computed when the compound is considered inactive[8], as these estimates are not applicable in such cases.

The *continuous hitcall*[9] is calculated based on the product of the probabilities of the following values [1]:

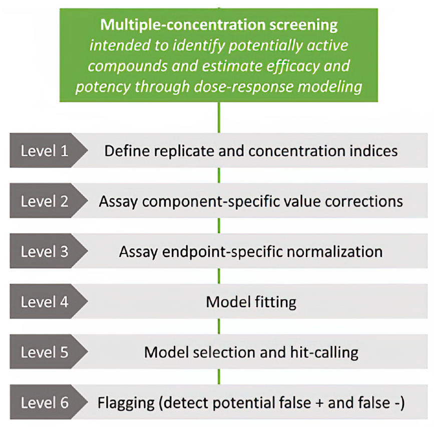
i. that at least one median response is greater than the efficacy cutoff computed by using the error parameter from the model fit and Student *t*-distribution to calculate the odds of at least one response exceeding the efficacy cutoff

[6] $AIC = -2\log(L(\hat{\theta}, y)) + 2K$

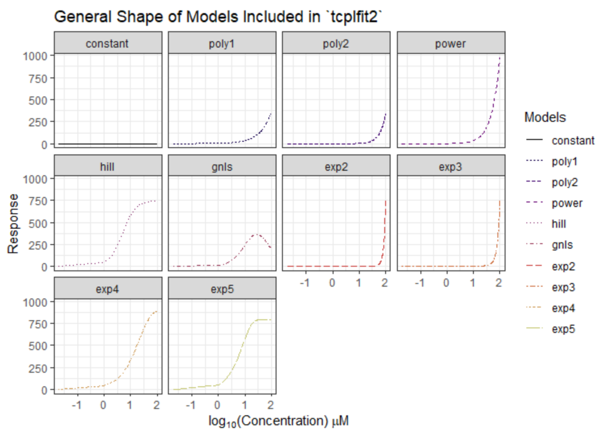
[7] The baseline region is defined as $0 + 3bmad$, where *bmad* represents the median absolute deviation calculated from the response values of the two lowest tested concentrations, where the assumption of the highest level of inactivity is legitimate.

[8] if the *winning* fit model was the constant model

[9] In legacy tcpl only binary hitcall was calculated


Multiple-concentration screening
intended to identify potentially active compounds and estimate efficacy and potency through dose-response modeling
Level 1 Define replicate and concentration indices
Level 2 Assay component-specific value corrections
Level 3 Assay endpoint-specific normalization
Level 4 Model fitting
Level 5 Model selection and hit-calling
Level 6 Flagging (detect potential false + and false -)

(a) Overview of tcpl's data analysis pipeline for multiple concentrations screening data.



(b) Employed curve-fit models in tcpl v3.0 for fitting concentration-response data series through the application of maximum likelihood estimation.

Figure 1.3: tcpl

ii. that the top of the winning fitted curve is above the cutoff:
the likelihood ratio of the one-sided probability of the efficacy cutoff being exceeded

iii. that the winning AIC value is less than that of the constant model:

$$\frac{e^{-\frac{1}{2}AIC_{\text{winning}}}}{e^{-\frac{1}{2}AIC_{\text{winning}}}+e^{-\frac{1}{2}AIC_{\text{cnst}}}}$$

Finally, after processing, each CRS is categorized into an appropriate fit category based on the level of certainty in the estimated bioactivity. Additionally, cautionary flags are assigned to account for problematic data series or uncertain fits and hits.

1.3 Data Overview

Presence Matrix

For the enhancement of data comprehension, a presence matrix denoted as $P \in 0, 1^{m \times n}$ is introduced. In this matrix, rows (indexed by i) represent assay endpoints a_i, and columns (indexed by j) indicate whether testing was conducted[10] (1) or not conducted (0) for compound c_j in those endpoints. Due to selective compound testing across different assay endpoints, matrix P is sparse.

[10]A compound is regarded as having been tested within an assay endpoint when there exists a corresponding concentration-response series accessible in the database.

For a visual representation of the presence matrix P covering all assay endpoints and compounds in *invitroDBv4.1*, refer to Figure 1.4.

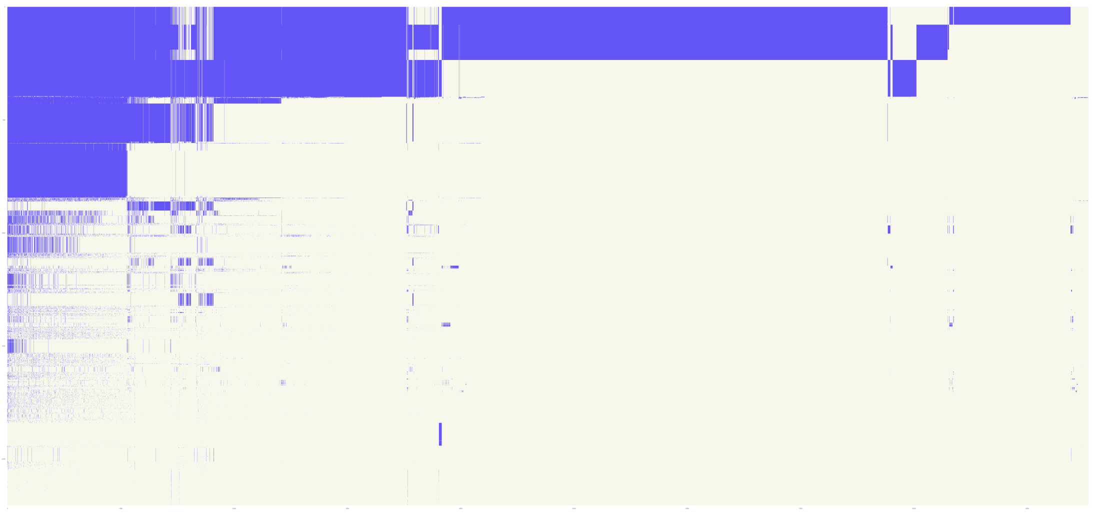


Figure 1.4: The *presence matrix* P covering all assay endpoints and compounds available in *invitroDBv3.5* with $m = 2205$ assay endpoints and $n = 9541$ compounds. The presence matrix is organized by sorting it based on the number of compounds present in each assay endpoint and the compounds are arranged in descending order of their presence frequency. The total count, where $P_{ij} = 1$, indicates the availability of 3 342 377 concentration-response series for downstream analysis.

Subsetting data

For this thesis only assay endpoints that have been tested with a minimum of 1000 compounds were exclusively taken into account. This criterion ensures the availability of sufficient data to train a machine learning model with a minimum level of robustness. Please consult Figure 1.5 to view a graphical depiction of the presence matrix P, which includes only the specific considerd subset of assay endpoints. From this point onward, this specific subset will be referred to as the *data* upon which the focus of this thesis will be directed.

In total, $\sum_{i,j} P_{ij} = 1\,372\,225$ concentration-response series are analyzed, encompassing a total of $\sum_{i,j} |S_{ij}| = 48\,861\,036$ concentration-response pairs across all compounds and assay endpoints.

1.4 pytcpl

This thesis introduces *pytcpl*[11], a streamlined `Python` package inspired by the `R` packages *tcpl* and tcplfit2, which were extensively discussed in Chapter **??**. The package

[11]https://github.com/rbBosshard/pytcpl

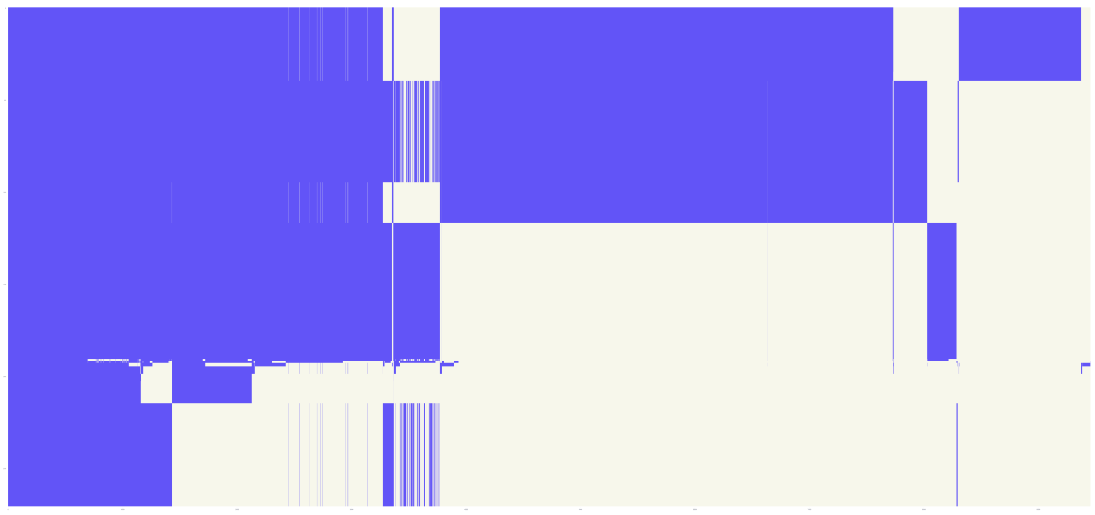


Figure 1.5: The *presence matrix* P covering only the subset of all of assay endpoints available in *invitroDBv3.5*, considered for this thesis, encompassing $m = 271$ assay endpoints and $n = 9456$ compounds. The total count, where $P_{ij} = 1$, indicates the availability of 1 372 225 concentration-response series for downstream analysis.

optimizes data storage and generates compressed *Parquet* files of the relevant raw data and metadata from *invitroDBv4.1*. This efficient strategy reduces storage needs, resulting in less than 10 GB within the repository, compared to the original 80 GB database. This obviates the need for a cumbersome, large-scale database installation, rendering downstream analysis more accessible and efficient. Our package is crafted to accomodate cusomizable processing steps and facilitate interactive data visualization with a novel *Curve Surfer*[12]. Furthermore, it enables researchers who prefer `Python` to easily participate in data analysis and exploration, overcoming any limitations associated with using `R` code.

1.4.1 Pipeline

The *pytcpl* pipeline is composed of four main steps, as illustrated in Figure **??**, summarising the steps also done by tcpl and tcplfit2.

1. Data collection
2. Cutoff determination and filtering (Meet conditions for curve fitting)
3. Curve fitting
4. Hit calling

[12]https://pytcpl.streamlit.app/

Model	Label	Equations[1]	Role in pytcpl
Exponential 3	exp3	$f(x) = a\left(\exp\left(\left(\frac{x}{b}\right)^p\right) - 1\right)$	Omit
Gain-Loss 2	gnls2	$f(x) = \frac{tp}{1+\left(\frac{ga}{x}\right)^p}\exp(-qx)$	New

[1] Model parameters: a: x-scale, b: y-scale p: (gain) power, q: (loss) power, tp: top, ga: gain AC50

Table 1.2: tcplfit2 Model Details

1.4.2 Cytotoxicity handling

We are determining cytotoxicity probability by assuming the ACC (or AC50, or other) has a Gaussian error distribution. We want to know the probability that:

$$ACC_{\text{target}} < ACC_{\text{cyto}}, \text{ which we'll call } P(\text{target action}).$$

We need:

$$\begin{aligned} &ACC_{\text{target}} \\ &ACC_{\text{cyto}} \\ \text{SD}_{ACC_{\text{target}}} - &\text{ we use 0.3 log units (this is in a few papers including the new ToxCast database one)} \\ \text{SD}_{ACC_{\text{cyto}}} - &\text{ this is harder to define. We could base on the MAD but I think we just use 0.3 log units he} \end{aligned}$$

Then we find:

$$\begin{aligned} P(\text{target action}) &= P(ACC_{\text{target}} < ACC_{\text{cyto}}) \\ &= P(ACC_{\text{cyto}} - ACC_{\text{target}} > 0) \end{aligned}$$

For this we use the R function `pnorm`:

$$\texttt{pnorm}(ACC_{\text{cyto}} - ACC_{\text{target}}, 0, \sqrt{\text{SD}^2_{ACC_{\text{target}}} + \text{SD}^2_{ACC_{\text{cyto}}}})$$

This approach needs to be implemented differently in two cases:

i) A matching viability assay is available: Then we use the ACC_{cyto} from the matching burst assay.

ii) A matching viability assay is not available, so we use the statistical approach. We use the median cytotox ACC for that chemical across all viability assays. Additionally, we account for $nhit$ and $ntested$. We can do this by stating:

$$P(ACC_{\text{cyto}} < \text{max tested concentration}) = \frac{nhit}{ntested}$$

and then:

$$P(\text{cytotox action}) = P(ACC_{\text{target}} > ACC_{\text{cyto}}) \cdot P(ACC_{\text{cyto}} < \text{max tested concentration})$$

and so:

$$P(\text{target action}) = P(ACC_{\text{target}} < ACC_{\text{cyto}}) \cdot (1 - \frac{nhit}{ntested})$$

1.4.3 Curve Surfer

Data visualization, overview of what is possible with the tool. Filter by assay endpoint, compound, etc.

1.5 Machine Learning Pipeline

1.5.1 Preprocessing

Subselecting the columns from the output tables generated by pytcpl: DTXSID identifier and continuous hitcall value. The feature inputs to the machine learning model is a molecular structure represented as fingerprint generated from a SMILES string uniquely determined by the compounds DTXSID identifier. The SMILES string is a linear representation of a compound's molecular structure. The SMILES string is converted to a molecular graph, which is then converted to a feature vector. The feature vector is then used to train a machine learning model. The machine learning model is then used to predict the hitcall value for a given compound. The machine learning pipeline is illustrated in Figure?.

1.5.2 Binary Classification

The goal is to predict whether a compound is active or inactive for a given assay endpoint. This can be formulated as a binary classification problem, where the input consists of the molecular structure fingerprint of the compound, and the output is the binarized hitcall value based on a specific decision threshold. The hitcall value is rendered to a binary variable, where 1 indicates that the compound is active and 0 indicates that the compound is inactive.

1.5.3 Regression

1.5.4 Massbank Validation

Bibliography

[1] T. Sheffield, J. Brown, S. Davidson, K. P. Friedman, and R. Judson, "tcplfit2: an R-language general purpose concentration–response modeling package," *Bioinformatics*, vol. 38, no. 4, pp. 1157–1158, Nov. 2021, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btab779. eprint: https://academic.oup.com/bioinformatics/article-pdf/38/4/1157/50422999/btab779.pdf. [Online]. Available: https://doi.org/10.1093/bioinformatics/btab779.

[2] C. for Computational Toxicology and U. E. Exposure, *Tcpl v3.0 data processing*, R package vignette for the tcpl package v3.0, CRAN, 2023. [Online]. Available: https://cran.r-project.org/web/packages/tcpl/vignettes/Data_processing.html.